6. Image Comparison

Program Name: Image.java Input File: image.dat

Bobby has made a new website. He originally had all of the images for his site on his computer. When he decided to host his site on the cloud, he had to transfer all his images to the cloud. Unfortunately, the cloud company changed the image format and names of the files to compress them, so now Bobby has to figure out which images are which. Unfortunately, with the compression algorithm, some of the pixel colors have changed slightly from the ones on his computer.

As a result, if Bobby wants to figure out which picture on the server corresponds to one on his computer, he needs to compute a difference, or "diff", between each image on the server, and the one on his computer. The correct picture on the server is the one with the smallest diff. The total diff score for the pair of images is the sum of the differences of each pixel. Each pixel is composed of a red, green, and blue channel, with a min value of 0 and a max value of 255 for each channel. Color values are traditionally stored in hexadecimal, or base 16, so they will range from 00 to FF. And as a result, one pixel consists of 6 hexadecimal digits, where the first two are the value of the red channel, the second two are the value of the green channel, and the third two are the value of the blue channel.

Input

The first line of input will be an integer T that is the number of cases to follow. The first line of input in each case contains three integers, N, W, and H, representing the total number of images on Bobby's site, and the width and the height of the images. All the images are of the same width and height.
2N images will follow. The first N images are the images from Bobby's computer, and the second N images are the images from the cloud server.
Each image consists of H + 1lines. The first line contains the filename of the image. The next H lines each contain W space-separated pixel values. (example: ffac1d dc23e6 b5ad4b)

Output

For each original image on Bobby's Computer, output the name of that image, followed by a colon and a space, followed by the name of the closest image on the cloud server. There will always be exactly one closest image. Place a blank line between each case.

Constraints

```
1 <= T <= 10
1 <= N <= 10
1 <= W, H <= 20
```

Example Input File

```
2
2 1 1
icon.png
ff89a3
background.png
666666
X3F.jpg
626163
AB.jpg
f190b2
3 2 2
circle.png
dd00ff c56a88
ee00ff 32684a
square.png
a3bcff ce3a82
21aafd 621aa1
triangle.png
3e00ff 4e29a8
3ea0ef 2684a3
5YaS.jpg
aabc3f c9318a
2aaf4 621aa5
H7rT.jpg
3a02f1 4e2aaa
39a1e1 2681a6
P77d.jpg
da00f9 c56b87
eb00fa 32604a
```

Example Output to Screen

```
icon.png: AB.jpg
background.png: X3F.jpg

circle.png: P77d.jpg
square.png: 5YaS.jpg
triangle.png: H7rT.jpg
```

Explanation of Output

To find which image is closest to icon.png, we do a diff between icon.png and X3F.jpg, and icon.png and AB.jpg, and see which image has the lower difference.

There is only 1 pixel, so the difference is just the difference of that pixel. For X3F.jpg, the difference is |0xFF-0x62| + |0x89 – 0x61| + |0xA3-0x63| = 157 + 40 + 64 = 261. For AB.jpg, the difference is |0xFF – 0xF1| + |0x89 – 0x90| + |0xA3 – 0xB2| = 14 + 7 + 15 = 36. 36 is less than 261, so AB.jpg is closer than X3F.jpg, and since there are only 2 images, AB.jpg is the closest. Therefore the closest image to icon.png is AB.jpg, and we can assume that AB.jpg is the corresponding image on the server.